说明：此 BIOS 程序已经通过精装板、豪华板上进行测试

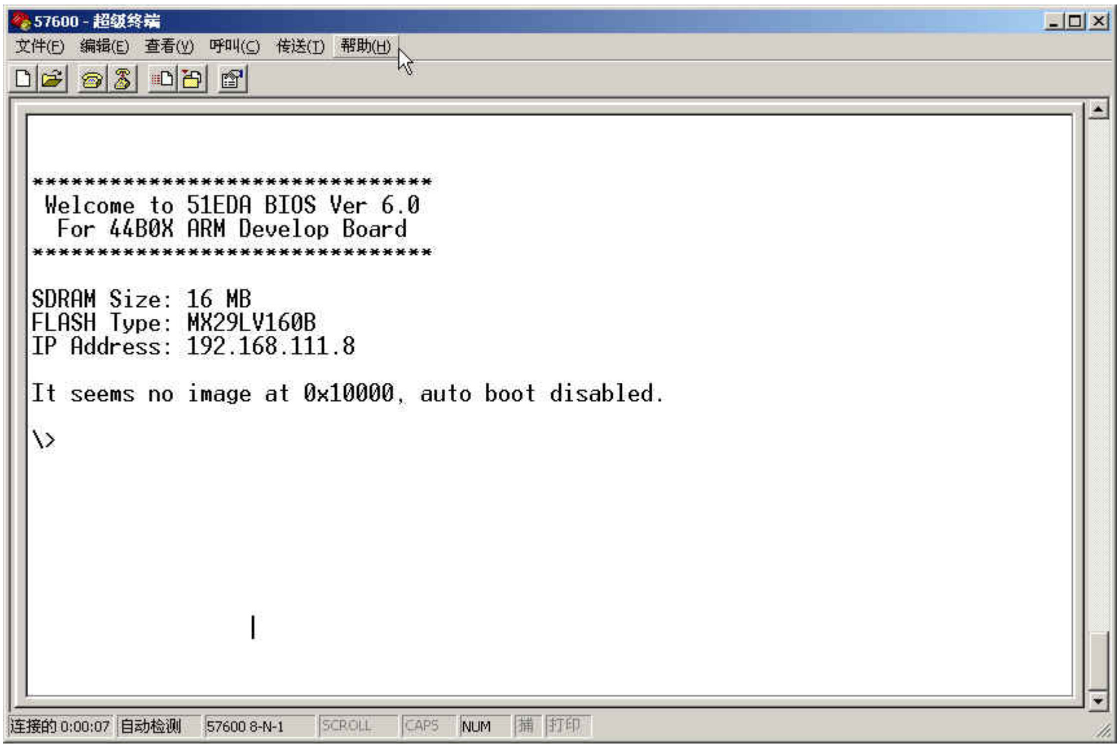
1- 配置串口

首先，在主机配置串口为：波特率 57600，数据位 8，奇偶校验无，停止位 1，流控无。
由于我们的 uClinux 的串口速率为 57600，因此为了兼容，本版的 BIOS 的波特率默认也改成了 57600。
下面以配置超级终端为例：

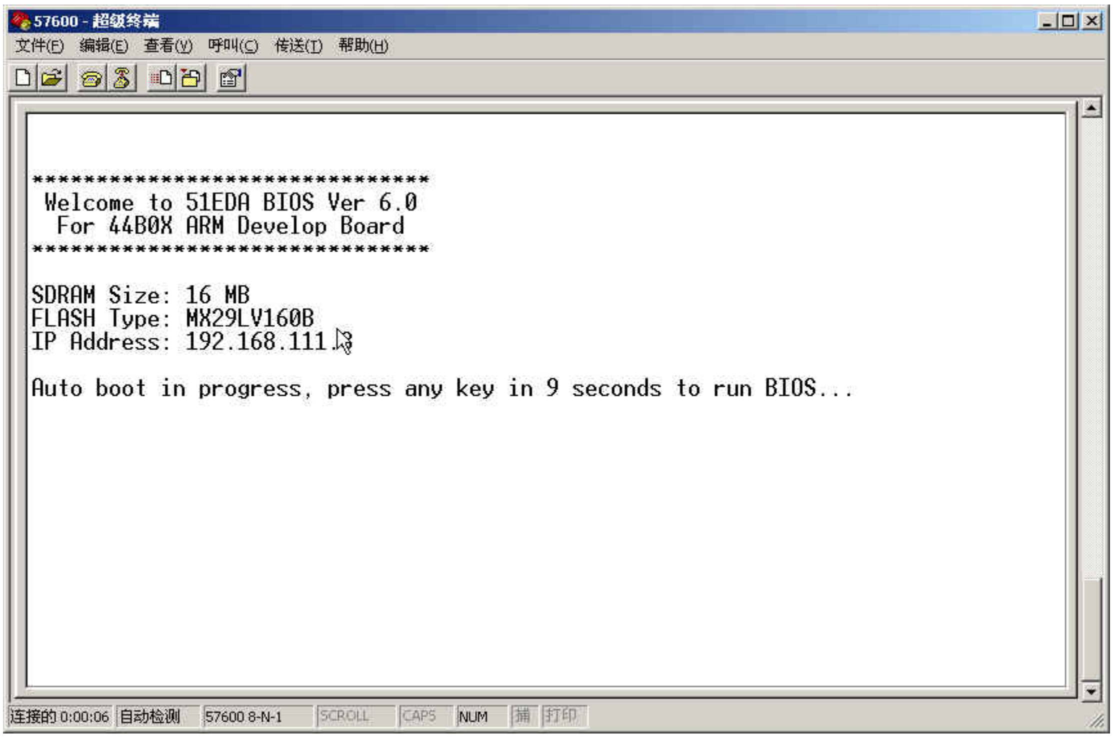
2- 烧写、使用 BIOS

烧写 BIOS 到 Flash 中（烧写方法见光盘：\演示、测试程序\使用 FLASHPGM 烧写程序演示动画）。如果在烧写的时候，进度条无反应，那么请先关闭烧写窗口，再按开发板上的复位按键（K1），然后再进行烧写。
烧写程序完毕之后，开发板会自动复位，BIOS 会自动探测你的 SDRAM 大小以及 NOR FLASH 的型号，并且给出相应的提示信息。之后，BIOS 首先判断 Flash 地址 0x10000 的内容，如果是 0xffffffff 的话说明是空的（Flash 没有写入代码）
对于 ARM 的指令来说，最高的 4bit 是条件码。如果条件码是 0xf，说明是 NV(Never)，这在 ARM 的指令中是不应该是用的。所以正好如果 Flash 是空的话那就是 0xffffffff，不是一条合法的 ARM 指令。以此可以判断 Flash 中是否写入了程序。
如果 Flash 的 0x10000 地址开始没有烧入用户的程序，那么 BIOS 直接进入命令行界面，

中国 EDA 技术网　本文欢迎转载，转载请注明出处　www.51eda.com



否则进入自动引导模式：



一开始设置等待时间为 10 秒。每 0.1 秒读取一次 UART，如果用户通过串口有输入就进入 BIOS，否则每 10 个 0.1 秒，等待时间减一，提示信息改一下，如果到了 0 秒还是没有按键，那就自动引导 0x10000 的程序。

如果有输入，那么也进入 BIOS 的命令行界面。

```
    |_|

For further information check:
http://www.uclinux.org/
ARM is your right choice!
Command:
Command: #/bin/getty -L ttyS0 57600 vt100
Execution Finished, Exiting

Sash command shell (version 1.1.1)
/>

********************************
 Welcome to 51EDA BIOS Ver 6.0
  For 44B0X ARM Develop Board
********************************

SDRAM Size: 16 MB
FLASH Type: MX29LV160B
IP Address: 192.168.111.8

Auto boot in progress, press any key in 6 seconds to run BIOS...

\>_
```

3- 烧写 image.rom（让 uClinux 在 NOR Flash 中运行）

执行 appprog（注意大小写）

```
********************************
 Welcome to 51EDA BIOS Ver 6.0
  For 44B0X ARM Develop Board
********************************

SDRAM Size: 16 MB
FLASH Type: MX29LV160B
IP Address: 192.168.111.8

It seems no image at 0x10000, auto boot disabled.

\>APPPROG
Bad command
\>APPPROG
Bad command
\>appprog
Now begin address for download, use address 0xc008000
Load image file from host
Type tftp -i 192.168.111.8 put filename at the host PC
Press ESC key to exit
```

通过网口下载已经编译好的 uClinux 的 ROM 文件，在执行下面操作时请必须保证与开发板连接的网卡地址为：192.168.111.XXX（当然你也可以通过修改 BIOS 程序改 IP 段----Main.c 中的 124 行）。

在主机端执行 TFTP 命令：

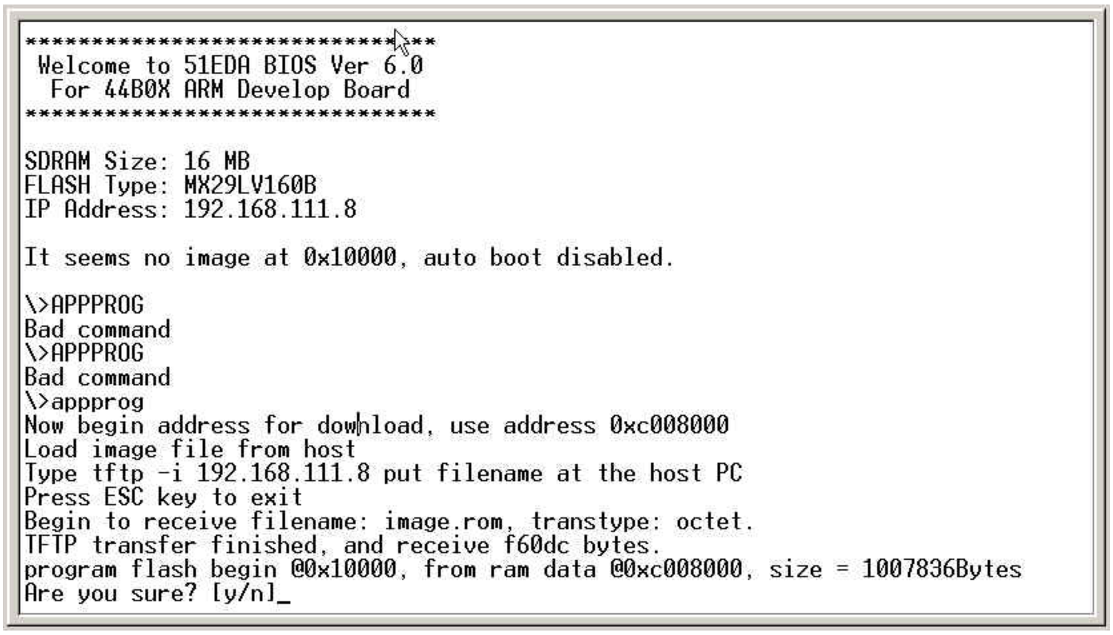
对于 Windows 系列，是 tftp –i 192.168.111.8 put image.rom

请注意 –i 参数表示使用二进制格式传输。如果没有加 –i 参数的话那么通过 TFTP 传输的数据就会出错。

如果是 Linux 用户，那么 TFTP 的使用和 Windows 系列又有所不同：

```
$ tftp 192.168.111.8
tftp> binary
tftp> put image.rom
Sent 1007836 bytes in 3.2 seconds
tftp>
```

在这里和 Windows 下面的 TFTP 一样，我们也需要设置成二进制的格式才能正确下载我们的镜像文件。



```
***********************************
 Welcome to 51EDA BIOS Ver 6.0
  For 44B0X ARM Develop Board
***********************************

SDRAM Size: 16 MB
FLASH Type: MX29LV160B
IP Address: 192.168.111.8

It seems no image at 0x10000, auto boot disabled.

\>APPPROG
Bad command
\>APPPROG
Bad command
\>appprog
Now begin address for download, use address 0xc008000
Load image file from host
Type tftp -i 192.168.111.8 put filename at the host PC
Press ESC key to exit
Begin to receive filename: image.rom, transtype: octet.
TFTP transfer finished, and receive f60dc bytes.
program flash begin @0x10000, from ram data @0xc008000, size = 1007836Bytes
Are you sure? [y/n]_
```

按 “ y ” 就开始烧写 Flash：

```
Flash Burn 0x050000 OK!
Flash Erase 0x060000 OK!
Flash Burn 0x060000 OK!
Flash Erase 0x070000 OK!
Flash Burn 0x070000 OK!
Flash Erase 0x080000 OK!
Flash Burn 0x080000 OK!
Flash Erase 0x090000 OK!
Flash Burn 0x090000 OK!
Flash Erase 0x0a0000 OK!
Flash Burn 0x0a0000 OK!
Flash Erase 0x0b0000 OK!
Flash Burn 0x0b0000 OK!
Flash Erase 0x0c0000 OK!
Flash Burn 0x0c0000 OK!
Flash Erase 0x0d0000 OK!
Flash Burn 0x0d0000 OK!
Flash Erase 0x0e0000 OK!
Flash Burn 0x0e0000 OK!
Flash Erase 0x0f0000 OK!
Flash Burn 0x0f0000 OK!
Flash Erase 0x100000 OK!
Flash Burn 0x100000 OK!
\>run 10000
```

输入 “ run 10000 ”，uClinux 就启动起来了。

```
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.255.255.0 lo
Command: ifconfig eth0 192.168.111.100 netmask 255.255.255.0 up
Command: mkdir /ramdisk/web
Command: cat /etc/motd
Welcome to
```

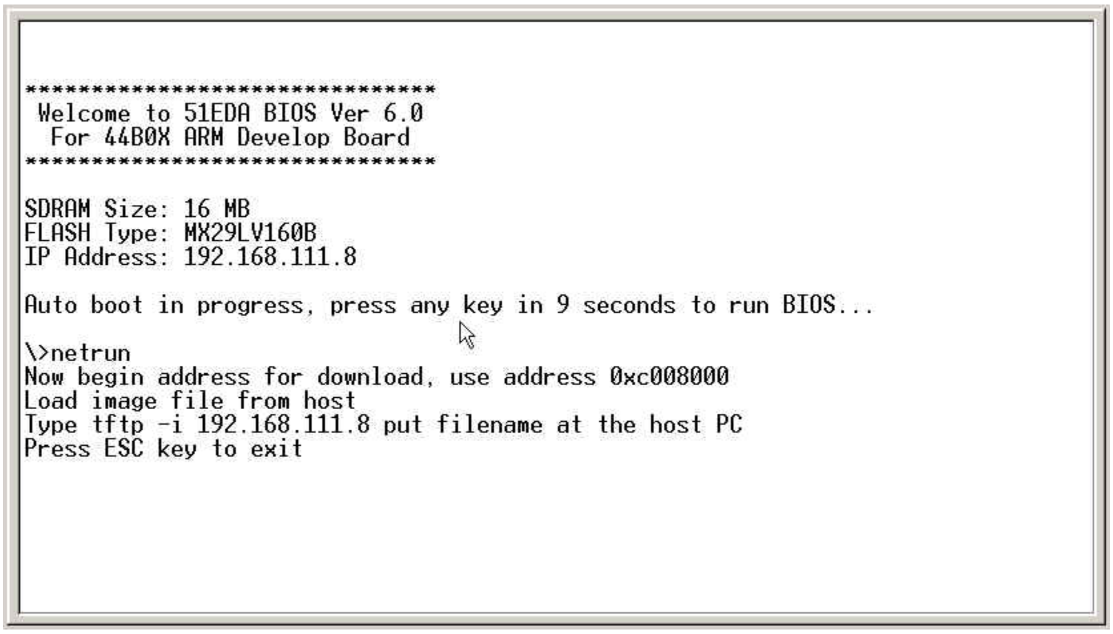
```
For further information check:
http://www.uclinux.org/
ARM is your right choice!
Command:
Command: #/bin/getty -L ttyS0 57600 vt100
Execution Finished, Exiting

Sash command shell (version 1.1.1)
/>
```

以上的方式是将 uClinux 在 Nor Flash 中运行，当关闭电源再次上电的时候 uClinux 会通过 BIOS 自动启动！

下面我们将介绍让 uClinux 在 SDRAM 中运行的方法-----------烧写 image.ram

4- 烧写 image.ram（让 uClinux 在 SDRAM 中运行）

烧写 BIOS 到 Nor Flash,打开终端，输入 netrun (注意大小写)：



通过网口下载已经编译好的 uClinux 的 ROM 文件，在执行下面操作时请必须保证与开发板连接的网卡地址为：192.168.111.XXX（当然你也可以通过修改 BIOS 程序改 IP 段----Main.c 中的 124 行)。

在主机端执行 TFTP 命令：

对于 Windows 系列，是 tftp –i 192.168.111.8 put image.ram

请注意 –i 参数表示使用二进制格式传输。如果没有加 –i 参数的话那么通过 TFTP 传输的数据就会出错。

如果是 Linux 用户，那么 TFTP 的使用和 Windows 系列又有所不同：

```
$ tftp 192.168.111.8
tftp> binary
tftp> put image.ram
Sent 2054584 bytes in 6 seconds
tftp>
```

在这里和 Windows 下面的 TFTP 一样，我们也需要设置成二进制的格式才能正确下载我们的镜像文件。

查看终端就可以发现，uClinux 已经启动了，以上的方式是将 uClinux 在 SDRAM 中运行，当关闭电源再次上电的时候 uClinux 不会启动，必须再重复操作一次 !!!

5- BIOS 常用命令：

appprog

从 TFTP 下载文件并且烧写到 0x10000 的位置（通常用于烧写 image.rom）

netrun

从 TFTP 下载 image.ram 到 0x0c008000 的位置并执行（通常用于 image.ram 的下载并直接运行）

run [地址]

从指定的地址开始执行（例如：run 10000）。默认的运行地址是 0x0c008000。地址的输入格式为 16 进制而不是 10 进制。

load [地址]

从 TFTP 下载文件到 RAM 指定地址的地方（默认 0x0c008000）。地址的输入格式为 16 进制而不是 10 进制。

prog <FLASH 地址> <RAM 地址> <长度>

从 RAM 地址烧写指定长度的数据到 FLASH 指定的地址

help

得到在线的提示消息，提示可用的命令

?

同 help

comload [地址]

从串口下载文件到指定的内存地址，在网络不通的情况下可以使用串口来传输数据，只是速度会比较慢。

comrun

从串口下载文件到内存 0x0c008000 处并执行。

md [地址]

察看内存的内容，例如下载的结果或者烧写的结果等

在这里我们列出了一些常用的命令。其他一些命令请参考 BIOS 的在线的帮助。

6- 如何从源文件编译

除了提供二进制的文件外，我们还提供了 BIOS 的源代码供用户进一步的学习和参考。

文件说明：

Net	：Rtl8019 驱动代码、网络协议代码
Src	：主程序代码
Inc	：主程序的头文件
51EDA_BIOS.mcp	：ADS1.2 工程文件
BIOS_51EDA.s19	：BIOS 的二进制文件，可以直接用 FlashPGM 烧写

注意事项：

1． Src 目录中的 44binit.s 文件于原开发板所带的其它原码中的 44binit.s 文件不同，不可被替换。
2． 工程中指定的 ro-base 为 0x0c7c000，不要改为 0，具体原来参考 44binit.s 中的注释。
3． 执行 load 命令后网络才会工作，平时是 ping 不通的。
4． 本 BIOS 目前支持 8M / 16M 的 SDRAM 自动识别，以及 SST39VF160 / MX29LV160T / MX29LV160B 这 3 种型号的 Flash。（注：MX29LV160T 的具体型号为 MX29LV160ATTC；MX29LV160B 的具体型号为 MX29LV160ABTC）

程序的说明：
本程序改编自 antiscle 公布的代码，在此首先对 antiscle 表示感谢。

hgxxx 替换的源代码的网络部分，对 shell 和 flash 也进行了一定的修改，使其可以在 51eda 的开发板上可以正常运行。

Cliff 针对 uClinux 的运行环境对串口波特率、系统主频进行了修改，并且提供了不同种类的 Flash 的识别，修正了原先程序在使用 16M SDRAM 时不能正确运行 uClinux 的情况，添加对 SDRAM 大小的检测。

欢迎广大网友投稿：www.51eda.com　　Email:　web@51eda.com